2008-01-14

Tentamen för TDP002 Imperativ programmering - teoridel

Skriv svaren på separat papper. Uppge namn och personnummer på varje inlämnat papper. Uppgifterna löses enskilt. Inga hjälpmedel. Kommunikation med andra kursdeltagare/utomstående under tentamenstillfället strikt förbjudet.

Betyg 3: 7 poäng.
Betyg 4: 11 poäng (5 allm / 6 python)
Betyg 5: 14 poäng (6 allm / 8 python)
Totalt: 20 poäng (12 uppgifter).

Både rätt innehåll och välformulerad text krävs för full poäng på uppgifterna.

Det totala betyget på tentamen är detsamma som det lägsta betyget av teoridel och problemdel.

Tid för teoridel: 2 timmar.

Allmän teori

Uppgift 1 (1p)**:** Vilka av följande påstående är rätt:

1. Med räckvidd menas hur omfattande effekten av en programsats är
2. Alla subprogram har sidoeffekter, synliga för användaren eller dolda
3. Anropsstacken används av interpretatorn för att hålla reda på anrop till subprogram
4. Subprogram används ofta för att undvika att program blir för abstrakta
5. Stegvis förfining kallas en metod för att gå ifrån problem till implementation

Uppgift 2 (1p): Ange två kontrollstrukturer som används i ett imperativt språk som t ex Python.

Uppgift 3 (1p): Vad är skillnaden mellan följande tre datornotationer: maskinspråk, assemblerspråk och högnivåspråk.

Uppgift 4 (1p): Definiera begreppet algoritm.

Uppgift 5 (2p): Ange decimal, binär och oktal kod för det hexadecimala talet AA. Varför är oktal och hexadecimal kod så bra för just datorprogrammering?

Uppgift 6 (2p): En sökningsalgoritm söker reda på positionen ett visst tal befinner sig i i en given sekvens. Vad krävs av denna sekvens för att detta ska gå att göra på O(log n)? Ange namnet på en sökningsalgoritm som har i värsta fallet komplexitet O(log n).

Uppgift 7 (2p): Betrakta tre följande programsatser i Python:

Fall 1:

```
if (x or y) and not z:
    u = 2
else:
    u = 4
```

Fall 2:

```
if (x and y) or not  z:
    u = 2
else:
    u = 4
```

Fall 3:

```
if (x and not z) or (y and not z):
    u = 2
else:
    u = 4
```

Vilket av föjande påstående stämmer:

1. Fall 1 och Fall 2 ger samma värde ut på u oavsett värde in på x, y och z
2. Fall 1 och Fall 3 ger samma värde ut på u oavsett värde in på x, y och z
3. Fall 2 och Fall 3 ger samma värde ut på u oavsett värde in på x, y och z
4. Fall 1, Fall 2 och Fall 3 ger alla samma värde på ut på u oavsett värde in på x, y och z

Python

Uppgift 8(1p): Vad är felet med följande syntax:

```
x = [1,2,3]
while x:
  if  x = [1,2]:
    break
  else:
    x = x[1:]
```

Uppgift 9(1p): Ange tre viktiga skillnader mellan datatyperna list och dictionary i Python.

Uppgift 10 (2p): Skriv ett Python-uttryck som gör en s k slice med de två första elementen ur en tupel som skapats genom att konkatenera en tupel som innehåller strängen hej och en tupel som innehåller talen ett, två och tre.

Uppgift 11 (2p): Förklara vad det innebär att funktioner är objekt i Python. Varför har man gjort språket sådant? Ge exempel på hur detta kan utnyttjas på ett bra sätt med ett kodexempel.

Uppgift 12 (2p): Vad skrivs ut i den interaktiva tolken efter detta:

```
>>> L1 = [3,9,2]
>>> L2 = [2, L1, 3]
>>> L1[1] = [2,3]
>>> L2
```

Uppgift 13 (2p): Vad skrivs ut när följande modul exekveras:

```
objects = [8,3]

def get_value():
   global value
   value = 3
   return value

def make_value():
   value = objects
   value.append(get_value())

value = 8
objects.append(8)

make_value()
print objects
print value
```